UNIFICANDO EL ANÁLISIS DE SISTEMAS Y LA PROGRAMACIÓN DE SOFTWARE PARA LENGUAJES ORIENTADOS A OBJETOS

jofrantoba

Kiongo Technology, Av. Manuel Seoane 761, Lambayeque – Perú

Resumen

En este artículo se realiza una propuesta para unificar el análisis de sistemas y la programación del sistema, mediante el uso de un patrón de diseño que encaja con las metodologías de desarrollo de software tradicionales y agiles.

Teniendo en cuenta que hasta ahora todos realizamos el análisis de sistemas con el objetivo de entender y documentar los requerimientos del cliente para su posterior programación, el analista de sistemas prepara fichas detalladas para que puedan ser leídas por los programadores con el objetivo de plasmar mediante algún lenguaje de programación los requerimientos funcionales del sistema.

Todo parece encajar en el ciclo de vida del software usando cualquier metodología, pero si vemos la parte técnica y práctica, el analista de sistemas brinda las fichas detalladas de los requerimientos funcionales al programador y espera que los requerimientos funcionen tal como él los entiende, el programador desarrolla las fichas y es aquí donde empieza a perderse la trazabilidad entre el análisis de sistemas y la programación, pues no tiene un patrón para desarrollar las fichas sin perder la referencia hacia el análisis de sistemas.

Se pretende brindar un marco para unificar el análisis de sistemas y la programación, con el objetivo de brindar una trazabilidad bidireccional entre ambas etapas, que permitan mapear los requerimientos funcionales tanto en el análisis como en la programación.

Palabras claves: Requerimientos funcionales, patrón de diseño, análisis de sistemas, programación.

1. Introducción

Unificar el análisis de sistemas con la programación para brindar una trazabilidad bidireccional entre ambas a partir de los requerimientos funcionales con lleva a realizar actividades puntuales que permiten relacionar ambas etapas. Todo empieza en el análisis del sistema, cuando se engloba funcionalidades atómicas en un paquete con un nombre genérico que permita identificarlas, los analistas usamos estos tipos de diagramas para limitar las funcionalidades que tendrá nuestro sistema, posterior a realizar toda la etapa de análisis y documentar las fichas de requerimientos funcionales que no dejan nada a la interpretación del programador, este empieza a desarrollar un patrón de diseño basado en MVC con una modificación con el objetivo de unificar el análisis con la programación la cual nombraremos como EPVC (entity– process – view - control), la cual sugiere que cada paquete que engloba requerimiento funcionales atómicos se convertirá en una Clase, donde cada método de la Clase es un requerimiento funcional dentro del análisis. Esto permitirá mapear de forma fácil la documentación de análisis cuando haya algún problema o simplemente se requiere comprobar la realización correcta del requerimiento funcional.

2. Unificando el análisis de sistemas con la programación

Aquí no vamos a explicar como se realizan las actividades del ciclo de vida del software haciendo uso de metodologías agiles o tradicionales, se pretende mostrar como lograr el mapeo y la trazabilidad entre el análisis y la programación, para ello es obligatorio realizar actividades que permitan relacionar el análisis con la programación. Por esta razón hemos dividido las actividades en actividades de análisis y actividades de programación para que cada equipo de desarrollo lo inserte según la metodología que usa para el ciclo de vida del software.

a. Actividades de Análisis

Elaborar diagrama de paquetes

Un diagrama de paquetes proporcionará el encapsulamiento de los requerimientos funcionales a ser desarrollados, además de saber que dicho nombre se convertirá en una Clase que contendrá métodos que referencien a cada requerimiento funcional.

En el siguiente diagrama de paquetes se ejemplifica:

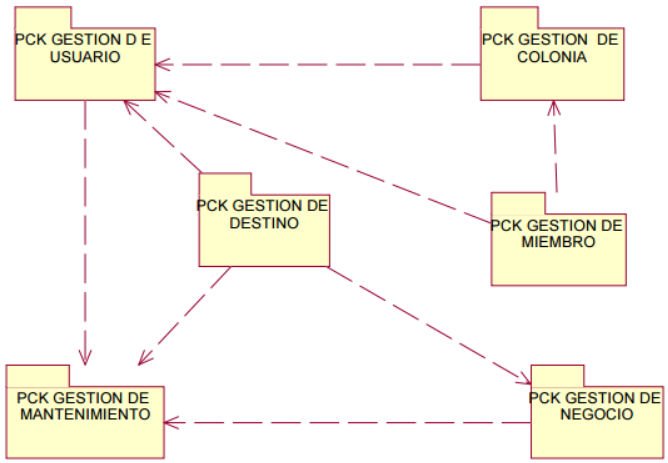

Ilustración 1: Diagrama de dependencia de paquetes

Las flechas apuntan hacia el paquete del cual son dependientes, en el ejemplo se observa que los paquete de Gestión de Destino no se puede desarrollar sin haber implementado los paquetes de Gestión de Mantenimiento, Usuario y Negocio.

Elaborar ficha de requerimientos funcionales

Dentro de cada paquete se empieza a ordenar todo los requerimientos funcionales que serán desarrollados, tener en cuenta que el nombre del requerimiento funcional se convertirá en un método dentro de la clase, citaremos un ejemplo basado en una metodología clásica como RUP.

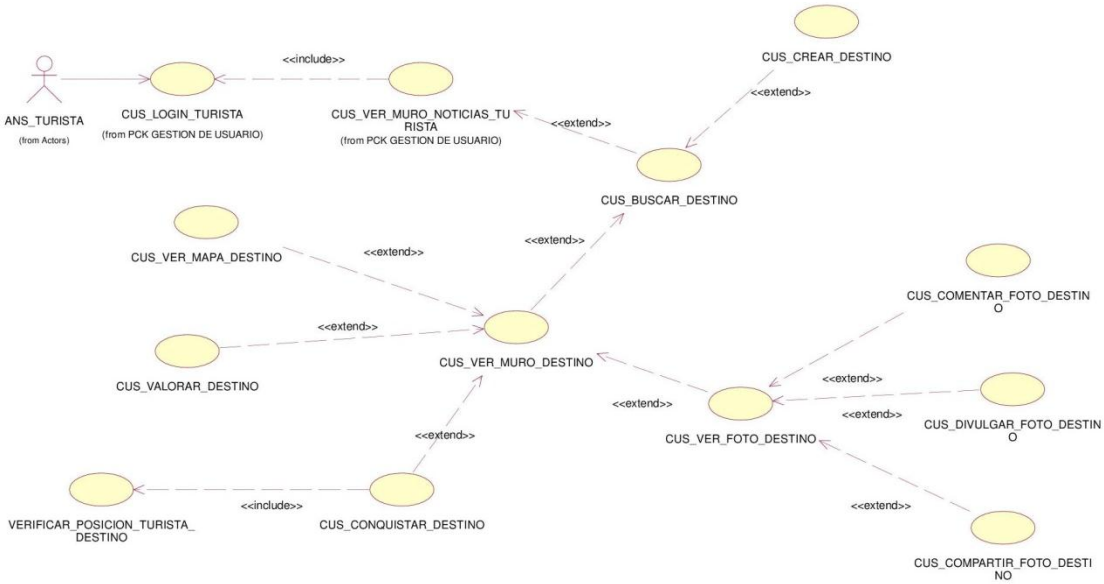

Ilustración 2: Diagrama de casos de uso de sistema - PCK Gestión Destino

El diagrama anterior no trata de obligar a usar la metodología RUP, solo muestra el encapsulamiento de funcionalidades dentro de un paquete. Dejando en claro que cada requerimiento funcional se convertirá en un método, por esta razón el programador debe colaborar al momento de dar nombres ya que los mismos nombres deben ser usados al momento de desarrollar.

Elaborar mockups del sistema

Es importante que analista y programador participen de esta actividad ya que no solo se debe coordinar los nombres que llevarán las interfaces sino también el flujo entre estas.

Diseñar base de datos

El diseño de base de datos se relaciona con el desarrollo de los beans o entidades en la programación, es fundamental que el desarrollador también participe del diseño del almacén de datos y participar en el nombramiento de la entidades, esto permitirá desarrollar la unicidad en el desarrollo de software

b. Actividades de programación

Estas actividades permitirán mapear los paquetes, los requerimientos funcionales, las entidades e interfaces con la programación, para poder realizar esto es necesario que los mismos nombres definidos en el análisis sean usados en la programación, las actividades claves es diseñar la arquitectura que usará y el patrón de diseño que se implementará en la programación.

Aquí solo hablaremos del patrón de diseño que permitirá la unificar ambas etapas, pues la arquitectura software y el lenguaje de programación no influyen sobre el patrón de diseño, ya que un patrón de diseño resuelve una problemática.

Patrón de diseño Entidad Process Control Vista (EPCV)

Para desarrollar la propuesta tomaremos a Java como lenguaje de programación.

- **ENTIDAD**

  La capa de entidad se centra en desarrollar todas las operaciones CRUD sobre las entidades individualmente, es una capa que se centra en la entidad y no piensa en los procesos de negocios que soportara el sistema, a cada entidad de base de datos corresponde un bean a

desarrollar con el igual o mayor número de atributos, ya que algunos atributos son usados como auxiliares dentro de la programación. En la siguiente imagen se nota que si en el almacén de datos existe una entidad “A”, en la programación existirá en bean, dao y logic “A”, es importante saber que en ninguna de estas capas se abre la conexión al almacén de datos, ya que desde la perspectiva técnica cuando se deseaba hacer un requerimiento funcional que guardaba en más de una entidad, se tenía que elegir en que Logic desarrollar la funcionalidad, este es el principal problema, ya que ningún Logic es más importante que otro, quizá solo habría que elegir alguna y escribir el requerimiento funcional ahí, pero esto genera desorden y pierdes la trazabilidad entre el análisis y la programación.

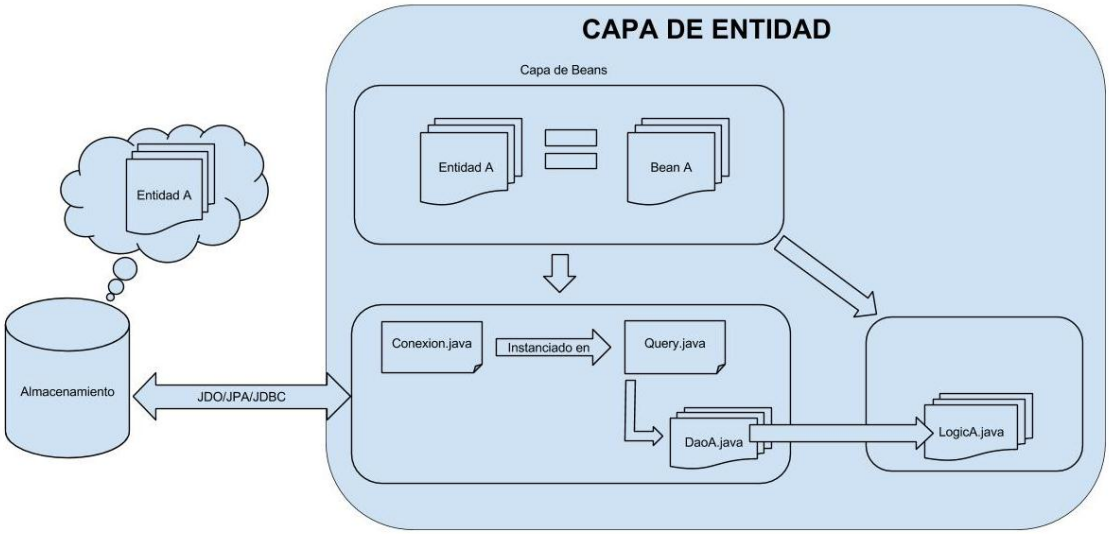

Ilustración 3: Capa de entidad

- **PROCESOS**

  Esta capa se relaciona con el diagrama de paquetes, ya que cada paquete se convertirá en un Clase en la programación que contendrá métodos que referencian a los requerimientos funcionales contenidos en los paquetes. En cada método de la Clase se abrirá la conexión al almacén de datos y se usara tantos logics como sean necesarios para guardar la data sobre las entidades que el requerimiento funcional ordene.

  En la siguiente imagen podemos notar que los beans y los logic serán usados dentro de esta capa de procesos, permitiendo de esta manera un uso óptimo de la conexión al almacén de datos ya que la conexión se apertura en cada método y se ira transfiriendo a cada logic y dao con el objetivo de almacenar data sobre las entidades que se requiera.

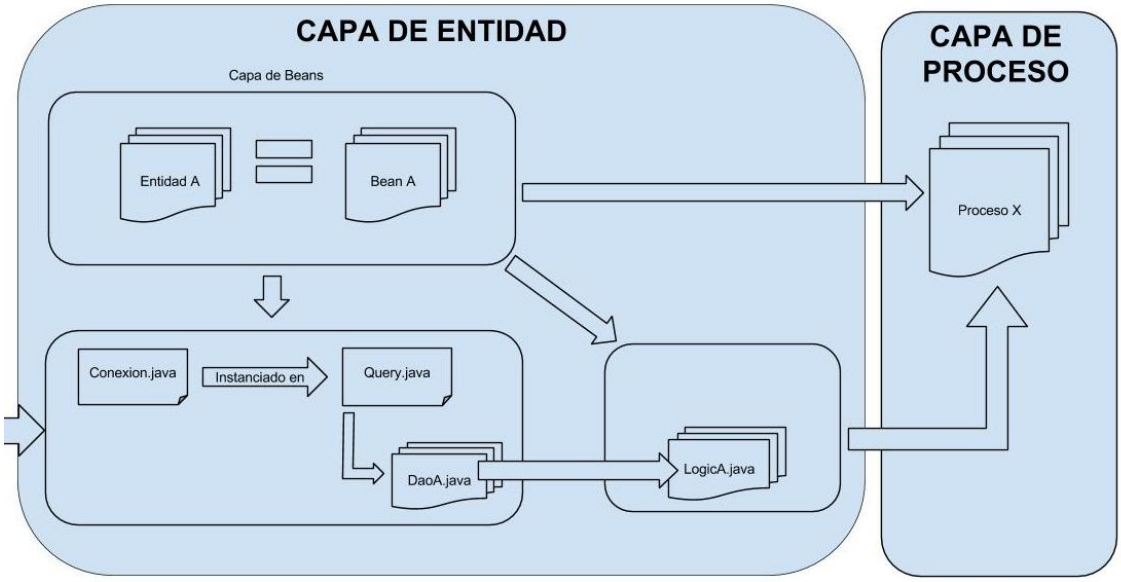

Ilustración 4: Capa de proceso

A partir de la capa de procesos podremos desarrollar el webservice para que sea interoperable con otras tecnologías, permitiendo acceder a la lógica de negocios y crear aplicaciones que solo consuman los servicios expuestos, en la siguiente figura se muestra la relación del webservice con la capa de procesos.

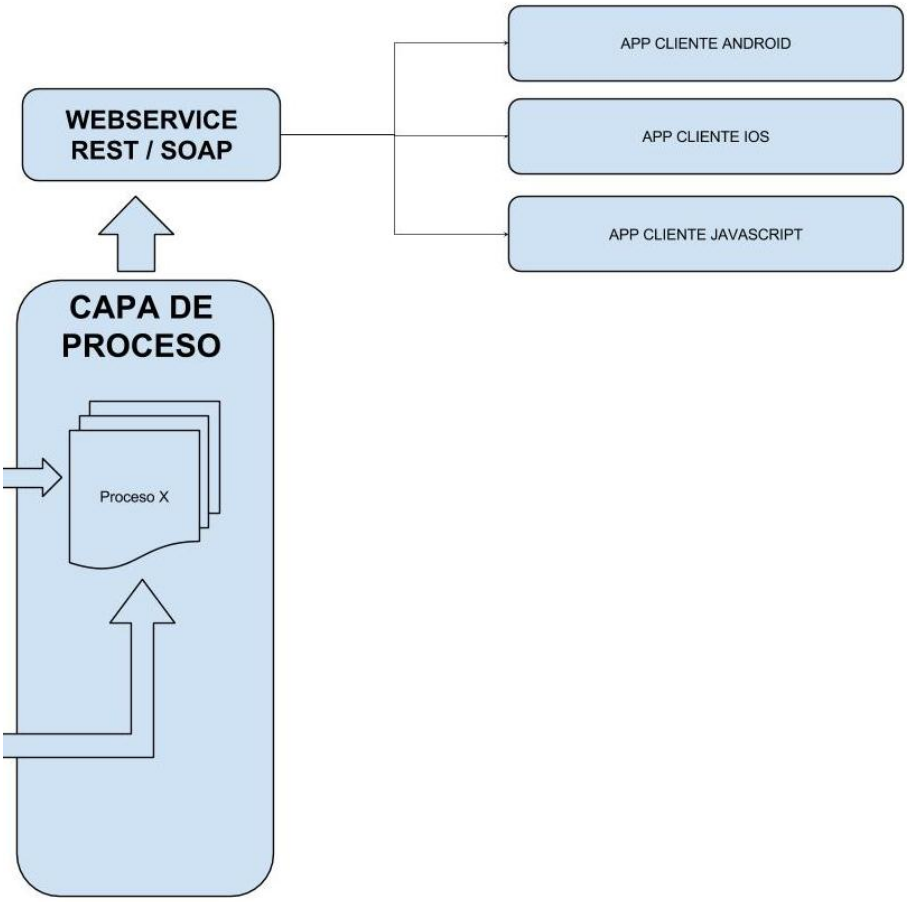

Ilustración 5: Desarrollo de webservice a partir de la capa de procesos

- **CONTROL**

El controlador es la capa que permitirá la comunicación cliente – servidor, usar esto en caso de aplicaciones web, si son aplicaciones desktop obviarlo y pasar a la capa de vista e integrar la vista con la capa de procesos. En la siguiente imagen se muestra la relación entre la capa de control que recibe una petición de un requerimiento funcional, el cual lo solicita a la capa de procesos y cuando la capa de proceso termina de procesar pasa la información al controlador para que dé una respuesta a la vista.

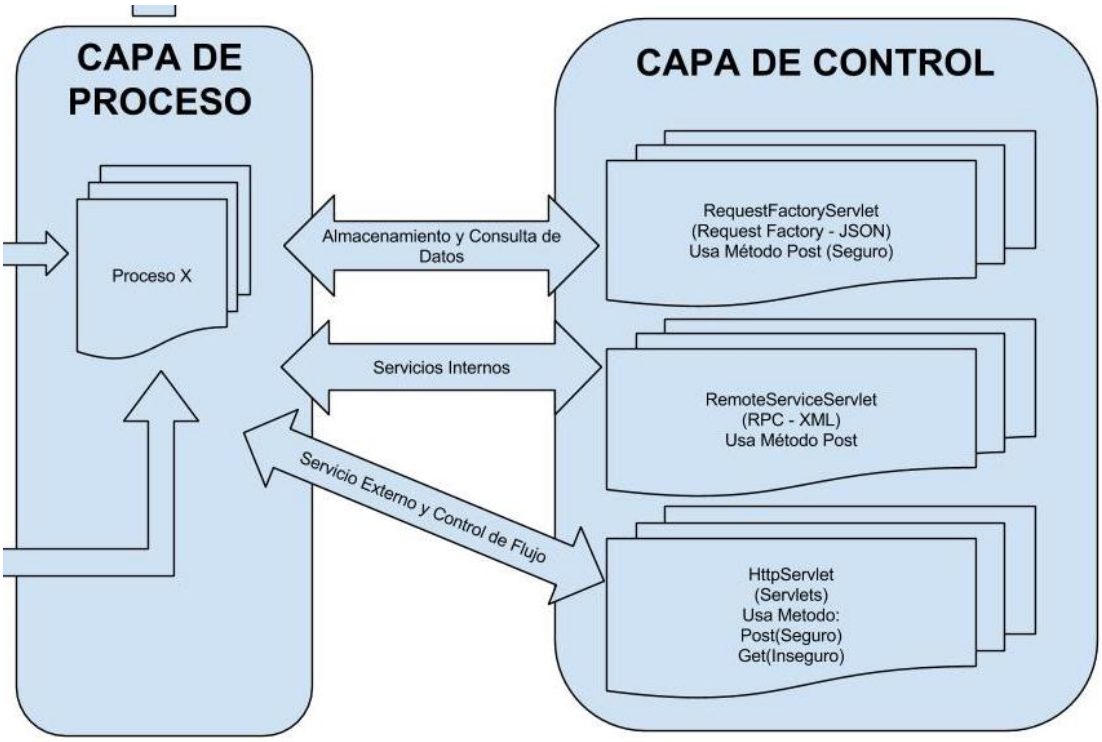

Ilustración 6: Capa de control

- **VISTA**

En la ilustración se muestra la vista mediante el uso de tecnología Java, pero se puede usar tecnología análoga para el desarrollo de la vista, ya que la capa de vista contiene todas las interfaces gráficas de plasmadas en los requerimiento del sistema, cuyos nombres deben ser los mismos que fueron planteados en el mockup.

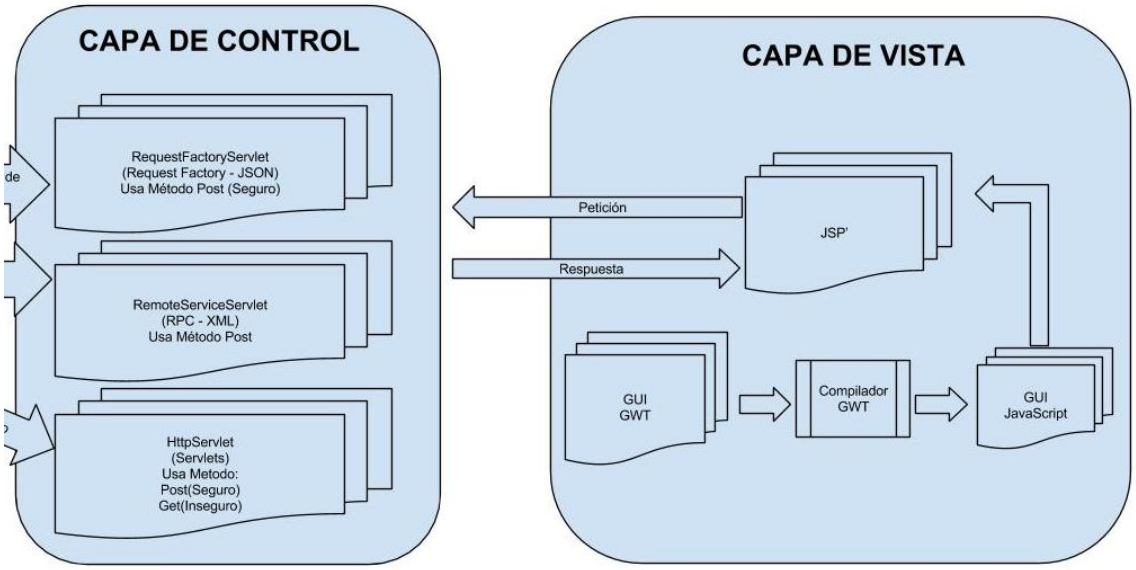

Ilustración 7: Capa de vista

Para el desarrollo de una interfaz gráfica web o escritorio se propone seguir este patrón, el cual muestra que cada interfaz debe ser un paquete o una carpeta que contenga tres archivos, el primero con el nombre del mockup que simplemente es una maqueta y donde cada botón ya sabe que método ejecutar cuando es presionado, un segundo archivo que permite nombrar los métodos de los requerimientos funcionales que se usarán en esa interfaz gráfica y los cuales serán llamada en la primera interfaz y vinculados al evento en cada control de la interfaz y por ultimo un tercer archivo que hereda de la interfaz maqueta, el cual sobreescribirá los métodos vinculados a los botones y permitirá la comunicación con la capa de control en el caso de aplicaciones web y con la capa de procesos en el caso de aplicaciones de escritorio. Este mismo patrón debería ser implementado al desarrollar aplicaciones móviles.

En la siguiente imagen se da un ejemplo de como se implementaría con tecnología Java.

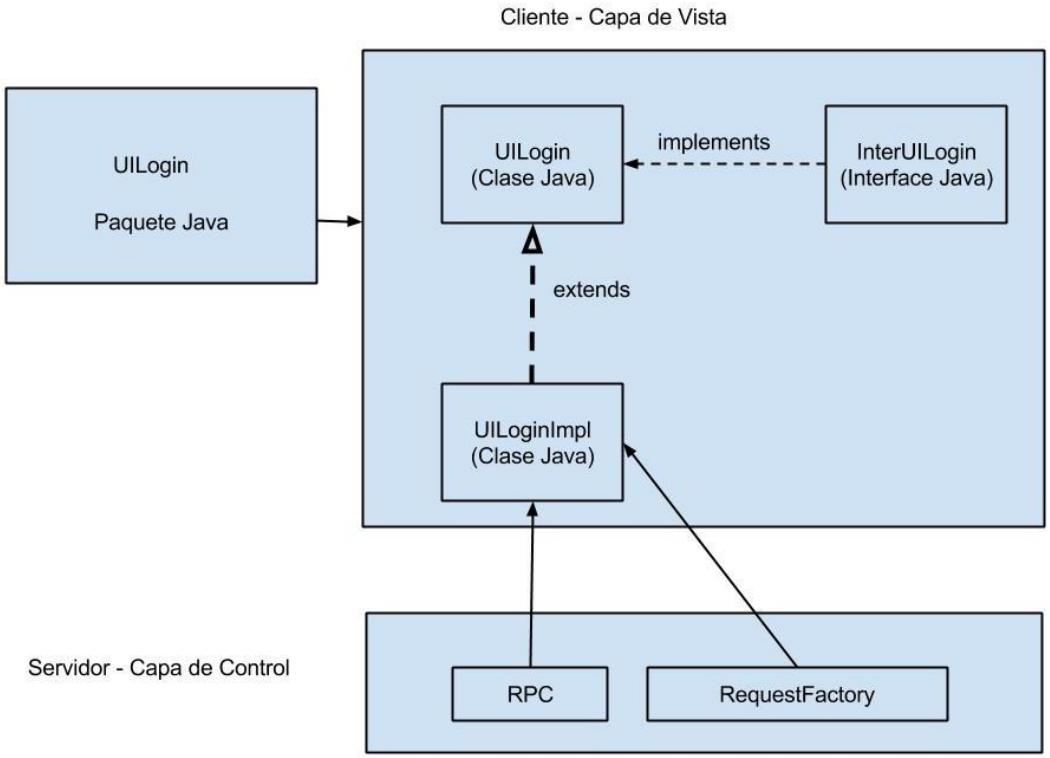

Ilustración 8: Patrón para crear interfaces gráficas